Intensification de la Recherche dans les Solveurs SAT Modernes

Said Jabbour[1] **Jerry Lonlac**[1,2] **Lakhdar Saïs**[1]

[1] CRIL - CNRS - Université Lille-Nord de France F-62307 Lens Cedex
[2] Département d'Informatique - Université de Yaoundé 1 B.P. 812 Yaoundé, Cameroun
{jabbour,lonlac,sais}@cril.fr

Abstract

Les Redémarrage et la recherche basée sur les activités sont deux composantes importantes et liées des Solveurs SAT Modernes. D'une part, la mise à jour des activités des variables impliquées dans l'analyse des conflits vise à circonscrire la partie la plus importante de la formule booléenne. Alors que le redémarrage permet au solveur de réorganiser les variables en focalisant la recherche sur la partie la plus importante de l'espace de recherche. Cette combinaison permet au solveur de diriger la recherche sur la sous-formule la plus contraignante. Dans ce papier, nous proposons de mettre en avant cette recherche basée sur l'intensification en collectant les variables rencontrées au cours de la denière analyse de conflit . A chaque redémarrage, ces variables sont à nouveau sélectonnées en utilisant l'ordre prédefini. Ce simple principe d'intensification apporte des améliorations significatives lorsqu'il est intégré aux solveurs SAT modernes.

1 Introduction

Le problème SAT, i.e., le problème de décider si une formule booléenne sous forme normale conjonctive (CNF) est satisfiable ou non est central en Informatique et en Intelligence Artificielle incluant les problème de satisfaction de contraintes (CSP), planification, raisonnement non-monotone, vérification de logiciels, etc. Aujourd'hui, SAT a gagné une audience considérable avec l'apparition d'une nouvelle génération de solveurs SAT capable de résoudre de très grandes instances issues du codage des applications du monde réel ainsi que par le fait que ces solveurs constituent d'importants composants de base pour plusieurs domaines, e.g., SMT (SAT modulo théorie), démonstration automatique, comptage de modèles, problème QBF, etc. Ces solveurs souvent appelés *Solveurs SAT Modernes* [10, 3], sont basés sur la procédure DPLL classique [2] renforcée avec : (i) une mise en œuvre efficace de la propagation unitaire à travers les structures de données supplémentaires et paresseuses (ii) stratégies de redémarrage [4, 8], (iii) activité basée sur l'heuristique de choix de variables (comme VSIDS) [10], et (iv) *apprentissage de clauses* [9, 10].

Les redémarrage et la recherche basée sur les activités sont deux composantes importantes et liées des Solveurs SAT Modernes. D'une part, la mise à jour des activités des variables impliquées dans l'analyse des conflits vise à circonscrire la partie la plus importante de la formule booléenne. Alors que le redémarrage permet au solveur de réorganiser les variables en focalisant la recherche sur cette sous-formule contraignante. Cette combinaison permet au solveur d'intensifier la recherche et en même temps d'éviter le trashing. Cette forte connexion bien connue entre les redémarrages et l'ordonnancement des variables a aussi une conséquence directe sur la clause apprise. Les effets de redémarrage sur la clause apprise ont été largement étudiés (e.g. [1, 7, 12]). Notre intuition est que si les solveurs SAT sont capables de résoudre efficacement des instances applicatives avec des millions de variables et de clauses, cela signifie que la partie la plus contraignante de la formule (ou sous-ensemble de variables) est de taille raisonnable. Ceci est lié à l'observation faite précédemment sur la taille des ensembles backdoor [13, 14] observé dans plusieurs domaines d'applications.

Dans nos travaux précédents, et dans le solveur parallèle porfolio ManySAT [6], nous avons montré comment les deux principes bien connus de diversification et d'intensification peuvent être combinés dans le contexte de l'architecture Maîtres/Esclaves [5]. Les Maîtres exécutent une stratégie de recherche originale,

en veillant à la diversification, tandis que les unités restantes, classées comme des esclaves sont là pour intensifier la stratégie de leur maître. Par intensification nous voulons dire que l'esclave explorerait différemment autour de l'espace de recherche exploré par le maître.

Dans ce papier, nous proposons de mettre en avant cette recherche basée sur l'intensification en collectant les variables rencontrées au cours de la dernière analyse de conflit. Ainsi, à chaque redémarrage, le solveur se branche en priorité sur ces variables. Ce simple principe d'intensification apporte des améliorations significatives lorsqu'il est integré au solveur SAT Minisat2.2.

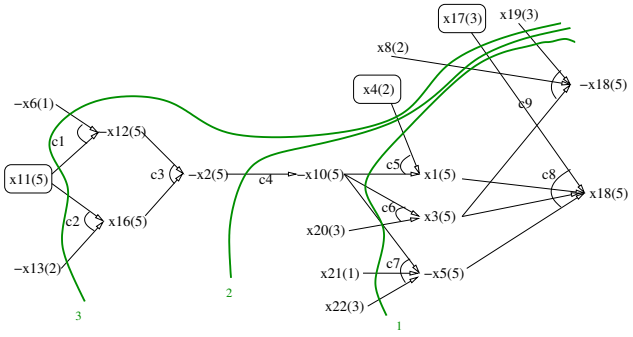

FIGURE 1 – Implication Graph $\mathcal{G}^{\rho}_{\mathcal{F}} = (\mathcal{N}, \mathcal{E})$

2 Intensification de la Recherche

Comme nous avons mentionné dans la section précédente, le lien entre les redémarrages et les heuristiques d'ordonnancement des variables joue un rôle important dans la résolution SAT. Ainsi, l'idée de base de ce papier est focalisée sur les relations entre ces deux composantes. En effet, au cours de la recherche, à chaque conflit, un parcours du graphe d'implications est effectué à partir du conflit jusqu'au dernier UIP (nœud x_{11} - coupe 3 dans Figure 1) et l'ensemble des variables correspondant aux différents nœuds visités sont insérées dans une queue. A chaque redémarrage, le solveur se branche en priorité sur les variables collectées. Lorsque toutes les variables de la queue sont affectées, le solveur suit l'heuristique de branchement ordinaire VSIDS [10]. De cette façon, les variables les plus proches du côté des conflits sont d'abord affectées en utilisant les polarités des littéraux progressivement sauvegardées [11].

Nous illustrons cette nouvelle approche de recherche basée sur l'intensification en utilisant un simple exemple.
Soit $\mathcal{G}^{\rho}_{\mathcal{F}}$ (Figure 1) un graphe d'implications associé à la formule CNF $\mathcal{F}$ et l'affectation partielle ρ. Chaque nœud dans le graphe d'implications correspond à une affectation d'un littéral de décision (nœud x_4, x_{17} et x_{11} affectés respectivement au niveau 2, 3 et 5) ou à un littéral assigné par propagation unitaire. Par exemple le littéral x_{16} est propagé grâce à la clause c_2 au niveau 5. Nous supposons que le dernier conflit apparu juste avant le redémarrage est représenté par ce graphe de conflit.

L'analyse de conflit qui est l'application de la règle de résolution à partir de la clause conflictuelle en utilisant les différentes implications implicitement encodées dans le graphe d'implications nous permet de visiter le nœud x_{11} qui est le dernier UIP. Au cours de cette analyse, l'ensemble des différents nœuds visités est $\{ x_{18}, \neg x_5, x_3, x_1, \neg x_{10}, \neg x_2, \neg x_{12}, x_{16}$ et $x_{11}\}$. Cet ensemble de variables est collecté dans la queue en suivant l'ordre selon lequel les différents nœuds sont visités. Au prochain redémarrage, le solveur se branche en priorité sur cet ensemble de variables.

3 Expérimentations

Nos tests furent effectués sur un cluster Intel Xeon quadcore avec 32GB de RAM à 2.66 Ghz. Pour chaque instance, nous utilisons un temps limite d'une heure. Nous utilisons un ensemble d'instances industrielles pris des compétitions SAT 2009 et 2011. Le nombre d'instances différentes correspond à 559.

MiniSAT avec notre stratégie ($MiniSat$ + $Intensification$) résoud 12 instances de plus que MiniSAT sans intensification. Le nombre d'instances résolues en plus est clairement significatif dans la résolution pratique de SAT. En effet, si on observe la récente compétition SAT 2011, le solveur classé premier résoud seulement 4 instances de plus que le solveur classé deuxième. Ces résultats obtenus par notre simple recherche basée sur l'intensification sont représentés dans la figure 2 et figure 3. Ils représentent les résultats des temps cumulés i.e le nombre d'instances (axe-x) résolues en un temps donné en secondes (axe-y). La figure 2 (respectivement figure 3), montre les résultats obtenus sur les instances satisifiables (respectivement insatisfiables). Les améliorations sont souvent plus importantes sur les instances insatisfiables.

Table 1 montre les résultats sur certaines familles d'instances SAT. Cette table montre des améliorations consistantes presque sur toutes les instances quand notre intensification est intégré à MiniSAT. Nous observons une croissance du nombre d'instances résolues. Par exemple, si nous considérons la famille goldb-heqc, (MiniSAT + Intensification) résoud 2 instances de plus que MiniSAT (les instances goldb-heqc-frg1mul et goldb-heqc-x1mul). Globalement, les améliorations

sont d'un ordre de magnitude (les instances velev-vliw-uns-4.0-9C1, 9dlx_vliw_at_b_iq1 et velev-pipe-sat-1.0-b10). Ces derniers résultats démontrent clairement que sur certaines familles, MiniSAT avec Intensification améliore clairement la version basique de MiniSAT. La table 1 confirme aussi que ces améliorations sont plus significatives sur les instances SAT insatisfiables.

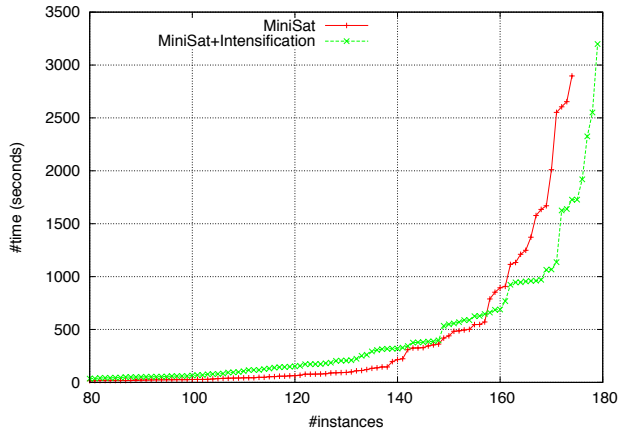

FIGURE 2 – Résultats sur les compétitions SAT 2009-2011 - instances Satisfiables - (Catégorie Industrielles)

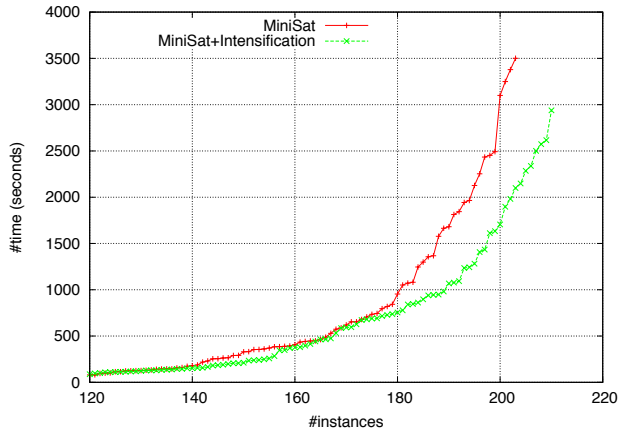

FIGURE 3 – Résultats sur les compétitions SAT 2009-2011 - instances Insatisfiables - (Catégorie Industrielles)

4 Remerciements

Ce travail a été soutenu par l'Agence Nationale de la Recherche - projet ANR programme blanc TUPLES.

Instance	SAT ?	MiniSat	MiniSat+I
goldb-heqc-alu4mul	N	140.94	128.64
goldb-heqc-term1mul	N	54.79	38.40
goldb-heqc-i10mul	N	187.06	129.14
goldb-heqc-dalumul	N	1663.95	181.31
goldb-heqc-frg1mul	N	–	715.66
goldb-heqc-x1mul	N	–	2500.59
velev-engi-uns-1.0-4nd	N	9.12	14.60
velev-live-uns-2.0-ebuf	N	18.53	9.84
velev-pipe-sat-1.0-b7	Y	21.51	23.51
velev-vliw-uns-4.0-9C1	N	3378.39	112.87
velev-pipe-o-uns-1.1-6	N	619.57	28.26
velev-pipe-o-uns-1.0-7	N	446.30	441.81
9dlx_vliw_at_b_iq1	N	1964.90	28.86
9dlx_vliw_at_b_iq2	N	–	70.07
9dlx_vliw_at_b_iq7	N	2642.42	1282.10
9dlx_vliw_at_b_iq3	N	–	189.83
9dlx_vliw_at_b_iq4	N	1750.56	283.74
9dlx_vliw_at_b_iq8	N	2293.63	1633.04
9dlx_vliw_at_b_iq9	N	2410.76	2572.97
9dlx_vliw_at_b_iq5	N	1631.90	464.47
9dlx_vliw_at_b_iq6	N	1267.90	840.68
velev-pipe-uns-1.0-8	N	–	414.21
velev-vliw-uns-4.0-9-i1	N	2095.14	592.52
velev-pipe-sat-1.0-b10	N	77.69	4.61

TABLE 1 – Résultats sur certaines familles d'instances industrielles

5 conclusion

Dans ce papier, nous proposons une nouvelle approche de recherche basée sur l'intensification. Cette recherche basée sur l'intensification est appliquée à chaque redémarrage du solveur SAT. Il collecte les variables rencontrées durant la dernière analyse de conflit. En utilisant un ordre prédéfini. Ces variables sont encore selectionnées jusqu'au sommet de l'arbre de recherche pendant le prochain redémarrage. Ce simple principe d'intensification donne des améliorations significatives quand il est intégré aux solveurs SAT état de l'art.

Notre motivation à l'origine de ce papier est de montrer qu'il reste encore des possibilitées l'améliorations des heuristiques d'ordonnancement des variables SAT. A notre connaissance, cette direction n'a pas retenue beaucoup d'attentions. Les améliorations obtenues par notre simple stratégie d'intensification suggère que des études plus poussées sur le lien entre les redémarrages, les heuristiques d'ordonnancement des variables et l'apprentissage sont nécessaires.

Références

[1] Armin Biere. Adaptive restart strategies for conflict driven sat solvers. In *International Conference,Theory and Applications of Satisfiability Testing, SAT'2008*, pages 28–33, 2008.

[2] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7) :394–397, 1962.

[3] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *International Conference,Theory and Applications of Satisfiability TestingSAT'2003*, pages 502–518, 2003.

[4] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proceedings of the National Conference on Artificial Intelligence, AAAI'98*, pages 431–437, Madison, Wisconsin, 1998.

[5] Long Guo, Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. Diversification and intensification in parallel sat solving. In *International Conference on Principles and Practice of Constraint Programming, CP'2010*, pages 252–265, 2010.

[6] Y. Hamadi, S. Jabbour, and L. Sais. ManySAT : a parallel SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 6 :245–262, 2009.

[7] Jinbo Huang. The effect of restarts on the efficiency of clause learning. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'2007*, pages 2318–2323, 2007.

[8] H. Kautz, E. Horvitz, Y. Ruan, C. Gomes, and B. Selman. Dynamic restart policies. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'02)*, pages 674–682, 2002.

[9] Joao P. Marques-Silva and Karem A. Sakallah. GRASP - A New Search Algorithm for Satisfiability. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 220–227, November 1996.

[10] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff : Engineering an efficient SAT solver. In *Design Automation Conference, DAC'01*, pages 530–535, 2001.

[11] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *International Conference,Theory and Applications of Satisfiability Testing, SAT'2007*, pages 294–299, 2007.

[12] Knot Pipatsrisawat and Adnan Darwiche. Width-based restart policies for clause-learning satisfiability solvers. In *International Conference,Theory and Applications of Satisfiability Testing, SAT'2009*, pages 341–355, June 2009.

[13] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'2003*, pages 1173–1178, 2003.

[14] R. Williams, C. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *International Conference on Theory and Applications of Satisfiability Testing, SAT'2003*, pages 222–230, 2003.